REMARKS

This paper is submitted in reply to the Office Action dated April 6, 2005, within the three-month period for response. Reconsideration and allowance of all pending claims are respectfully requested.

In the subject Office Action, claims 38-39 were rejected under 35 U.S.C. § 101 as being directed to non-statutory subject matter. In addition, claims 2, 19-20, 22 and 38-39 were rejected under 35 U.S.C. § 112 second paragraph. Furthermore, claims 1-9, 18, 21-28 and 37-39 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 4,183,083 to Chatfield (Chatfield) in view of U.S. Patent No. 6,202,199 to Wygodny et al. (Wygodny et al.), and claims 10-17, 19-20 and 29-36 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chatfield in view of Wygodny et al. and further in view of U.S. Patent No. 5,972,909 to Wilner et al. (Wilner et al.).

Applicants respectfully traverse the Examiner's rejections to the extent that they are maintained. Applicants have nonetheless amended claims 1, 2, 19, 20, 38 and 39 in deference to the Examiner. Applicants respectfully submit that no new matter is being added by the above amendments, as the amendments are fully supported in the specification, drawings and claims as originally filed.

First in the subject Office Action, claims 38-39 have been amended to expressly recite that the program code is computer executable or readable. Moreover, "the program code claims are patentable subject matter under 35 U.S.C. § 101 and must be examined," as held in *In re Beauregard*, 53 F.3d 1583 (1995).

Regarding the § 112 rejections, claims 2, 19 and 20 have been amended to address any perceived lack of clarity.

Now turning to the art-based rejections in the subject Office Action, and specifically to the rejection of independent claim 1, this claim generally recites a method of analyzing program execution within an operating system of a multithreaded environment. The method includes accumulating diagnostic data pertaining to a thread

accessing a resource, and storing the diagnostic data within a data structure at a location in the data structure correlated to the resource. The execution of a thread is predicated upon the thread's access to the resource within the multithreaded environment.

Prior to embodiments of Applicants' invention, locating a problematic task and/or resource within a multithreaded environment was a daunting and painstaking undertaking that relied on trace switch and other conventional programs. Such prior art programs generated preclusively voluminous amounts of data through which a programmer had to sift. The conventional programs are hindered by the nature of a multithreaded environment, where unlike in other computer systems, threads must share a relatively complex priority scheme that partially coordinates allocation of processing cycles as between different threads. Conventional debugging techniques consequently required tracking an amount of data that is commensurate with the exponentially high activity of the environment's multiple threads. Thus, the very efficiencies afforded by such multithreaded environments require special debugging programs, and prior to Applicants' embodiment, such programs generated data that was too overwhelming and disconnected to be useful.

The above problems with the prior art that are solved by the processes of claim 1 are exemplified by the cited prior art. That is, the Office Action admits on page 4 that the primary reference, Chatfield, fails to disclose a thread, let alone multiple threads. There is consequently neither a motivation nor a teaching in Chatfield to address debugging problems associated with the processes of multiple threads.

To remedy this deficiency, the Office Action asserts that it would have been obvious to combine the non-multithreaded processes of Chatfield with the multithreaded processes of Wygodny et al. However, Wygodny et al. actually teaches away from such a combination at column 2, lines 33-36. That is, Wygodny et al. acknowledges the additional complexities inherent to multithreaded environments that render useless those debugging programs (such as are disclosed in Chatfield) that are not designed for a

Page 9 of 13
Serial No. 10/022,982
Amendment and Response dated July 6, 2005
Reply to Office Action of April 6, 2005
IBM Docket ROC920010097US1
WH&E IBM/195
K:\ibm\195\Amendment and Response re 4-6-05 OA.wpd

multithreaded environment. There is consequently no motivation to combine Chatfield with Wygodny et al.

Even if such a hypothetical combination was motivated, in any case, the resultant combination would still fail to suggest accumulating diagnostic data pertaining to a thread accessing a resource, and storing the diagnostic data within a data structure at a location in the data structure correlated to the resource. Wygodny et al. relies on the trace algorithms described in the background of Applicants' invention at page 5. These conventional trace algorithms (col. 2, lines 59-67) require a programmer to sift through screens of data. Notably, data in Wygodny is never stored in a data structure correlated to a resource. The programmer using the conventional trace algorithm disclosed in Wygodny et al. is relegated to searching through volumes of material based only on the source code and elements ("other functions, processes" - col. 10, line53), i.e., not a resource. Similarly, there is no teaching or suggestion within Chatfield to store diagnostic data within a data structure correlated to a resource. The text cited by the Examiner, for instance, discloses storing merely either program or resource data, but does not teach or suggest correlating the two types of data, let along correlating a bucket storing the data to the resource. This correlation provides enormous efficiencies for programmers attempting to focus a debugging process within a multithreaded environment.

Applicants respectfully submit that independent claim 1 is consequently non-obvious over the prior art cited by the Examiner. Reconsideration and allowance of independent claim 1, as well as of claims 2-18 which depend therefrom, are therefore respectfully requested.

Next turning specifically to the rejection of independent claim 19, this claim generally recites a method of analyzing program execution within a computer system having a plurality of threads accessing a plurality of resources. The method includes calculating a time increment reflective of a duration a thread of the plurality of threads

waits for access to a resource of the plurality of resources, and storing the time increment within a bucket of a plurality of buckets comprising a hash array, each bucket being correlated to the resource. The execution of the thread is predicated upon the thread's access to the resource.

As discussed above in connection with claim 1, the combination of Chatfield with Wygodny et al. used to reject claim 19 is unmotivated. Similarly, Wygodny et al. teaches at column 2, lines 33-36 against a combination with Wilner et al., which like Chatfield, fails to contemplate a multithreaded environment. There is consequently no proper motivation to combine the references. However, even a hypothetical combination of the three references would still fail to suggest storing the time increment within a bucket of a plurality of buckets comprising a hash array, each bucket being correlated to the resource. Wygodny et al. and Chatfield do not suggest correlating data, let alone a time increment, in a bucket that is correlated to a resource. Wygodny et al. merely displays program data in the manner of a conventional trace algorithm, and Chatfield does not correlate resources to a hash bucket. Although Wilmer et al. mentions "time stamps," it does not calculate a time increment reflective of a duration of a thread (nor does it even contemplate a thread), and Wilmer furthermore provides no remedy to the above deficiencies relating to an absence of storing the time increment within a bucket of a plurality of buckets comprising a hash array, each bucket being correlated to the resource.

Applicants respectfully submit that independent claim 19 is novel and non-obvious over the prior art cited by the Examiner. Reconsideration and allowance of independent claim 19, as well as of claim 20 which depends therefrom, are therefore respectfully requested.

Next turning specifically to the rejection of independent claim 21, this claim generally recites an apparatus that includes at least one processor configured to execute a plurality of threads, a memory and program code resident in the memory that is configured to execute on the at least one processor. The program code is also configured

to accumulate diagnostic data pertaining to a thread accessing a resource. The execution of a thread is predicated upon the thread's access to the resource. The program code is further configured to store the diagnostic data within a data structure at a location in the data structure correlated to the resource.

Applicants respectfully submit that independent claim 21 is novel and non-obvious over the prior art cited by the Examiner for reasons similar to those presented with regard to claim 1. Specifically, the combination of Chatfield and Wygodny fails to disclose or suggest the concept of storing diagnostic data within a data structure at a location in the data structure correlated with a resource, in combination with the other features recited in the claim. Reconsideration and allowance of independent claim 21, as well as of claims 22-37 which depend therefrom, are therefore respectfully requested.

Finally turning specifically to the rejection of independent claim 38, this claim generally recites a program product that includes program code for analyzing program execution within an operating system of a multithreaded environment, wherein the program code is configured to accumulate diagnostic data pertaining to a thread accessing a resource, the execution of a thread being predicated upon the thread's access to the resource, and to store the diagnostic data within a block of the memory correlated to the resource, and a signal bearing medium bearing the program code.

Applicants respectfully submit that independent claim 38 is novel and non-obvious over the prior art cited by the Examiner for reasons similar to those presented with regard to claim 1. Specifically, the combination of Chatfield and Wygodny fails to disclose or suggest the concept of storing diagnostic data within a data structure at a location in the data structure correlated with a resource, in combination with the other features recited in the claim. Reconsideration and allowance of independent claim 38, as well as of claim 39 which depends therefrom, are therefore respectfully requested.
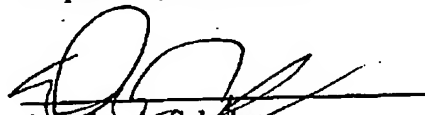
In summary, Applicants respectfully submit that all pending claims are novel and non-obvious over the prior art of record. Reconsideration and allowance of all pending

claims are therefore respectfully requested. If the Examiner has any questions regarding the foregoing, or which might otherwise further this case onto allowance, the Examiner may contact the undersigned at (513) 241-2324. Moreover, if any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

7/6/05

Date

Douglas A. Scholer
Reg. No. 52,197
WOOD, HERRON & EVANS, L.L.P.
2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
Telephone: (513) 241-2324
Facsimile: (513) 241-6234

Page 13 of 13
Serial No. 10/022,982
Amendment and Response dated July 6, 2005
Reply to Office Action of April 6, 2005
IBM Docket ROC920010097US1
WH&E IBM/195
K:\Ibm\195\Amendment and Response re 4-6-05 OA.wpd